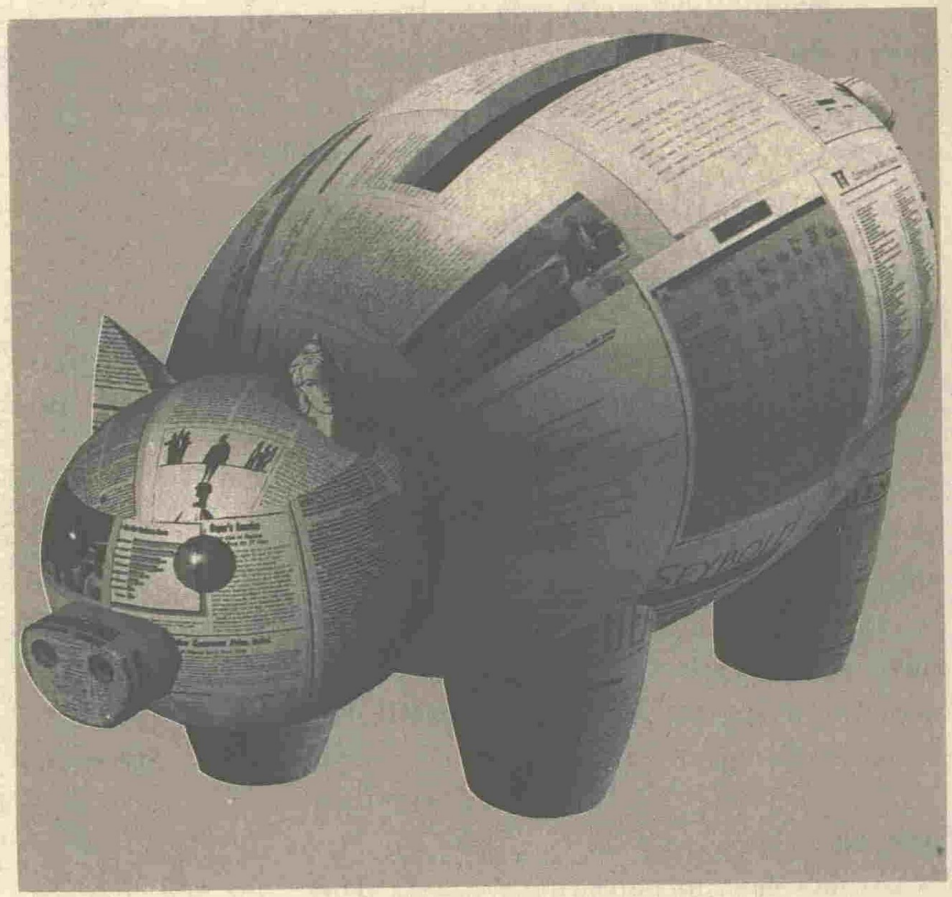
โอโอพี
: แฟ้มข้อมูลและสกุลรวม

โอโอพีจะช่วยให้คุณ สร้างแฟ้มข้อมูล เพื่อดำเนินการ กับข้อมูลได้ทุกแบบ



งานหลักของการใช้งานคอมพิวเตอร์โดยทั่วไปคือ การป้อน การเก็บ และการนำข้อมูลออกมาใช้ ในตอนที่แล้วผมได้กล่าวถึงแบบกรอกข้อมูล ทำให้สามารถป้อนข้อมูลได้ทุกรูปแบบ (ที่ไม่ซับซ้อนมากนัก) แต่เมื่อเก็บลงไฟล์ยังต้องเก็บในแบบไฟล์ของเรคอร์ดอยู่ ในฉบับนี้โอโอพีจะช่วยให้คุณสร้างแฟ้มข้อมูลเพื่อดำเนินการกับข้อมูลได้ทุกแบบ และเมื่อรวมกับแบบกรอกข้อมูลอาจกล่าวได้ว่า ต่อไปนาน ๆ เข้าจะพากันไม่รู้จักคำว่า RECORD ในภาษาปาสคาล

ไฟล์กับไฟล์ของเรคอร์ด

แต่ก่อนที่จะกล่าวถึงแฟ้มข้อมูล ผมเห็นว่าควรนำเรื่องราวที่เกี่ยวกับไฟล์มากล่าวถึง เพื่อแสดงให้เห็นว่าเราจะอ่านเขียนกับไฟล์ของเรคอร์ด โดยไม่ต้องรู้จักแบบข้อมูลของเรคอร์ดได้อย่างไร กล่าวคือ

ไฟล์ของเรคอร์ดก็คือ ไฟล์ไม่กำหนดแบบ (untyped file) ที่คอมพายเลอร์ดำเนินการบางอย่างแทนเรา เช่น เมื่อกำหนดแบบข้อมูลเรคอร์ด ตัวแปรไฟล์ของเรคอร์ด และไฟล์ไม่กำหนดแบบไว้ตามรูปที่ 1 ในการเขียนโปรแกรมตามปกติเมื่อใช้คำสั่งว่า

```
Reset (perfi);
```

คอมพายเลอร์จะดำเนินการเทียบเท่ากับการใช้คำสั่งว่า

```
Reset (dfi, SizeOf (oneper));
```

และเมื่อใช้คำสั่งว่า

```
Read (perfi, oneper);
```

จะเทียบกับการใช้คำสั่งว่า

```
BlockRead (dfi, oneper, 1, res);
```

ซึ่ง 1 หมายถึงจำนวนเรคอร์ดที่อ่านให้แก่ oneper ส่วนคำสั่ง res คือจำนวนเรคอร์ดที่อ่านได้จริง นั่นคือ เราสามารถที่จะอ่านเขียนกับไฟล์ของเรคอร์ดได้โดยเพียงแต่ทราบขนาดของเรคอร์ดเท่านั้น ส่วนการดำเนินการกับฟิลด์ก็สามารถดำเนินการได้โดยไม่ต้องกำหนดแบบข้อมูลเรคอร์ด ดังกล่าวในตอนที่แล้ว

```
TYPE
  AdrRcrdT = RECORD
    adrID   : String[9];
    name    : String[39];
    streetadr : String[69];
    city    : String[19];
    ZIP     : String[9];
    phone   : String[11];
              END;

VAR   perfi : FILE OF AdrRcrdT;  dfi : File;
      oneper : AdrRcrdT;  res : Word;
```

รูปที่ 1 เปรียบเทียบไฟล์ตามแบบข้อมูลกับไฟล์ไม่กำหนดแบบข้อมูล

แฟ้มข้อมูล

หลักการอันหนึ่งของการเขียนโปรแกรมในแบบโอโอพี คือเป็นการทำแบบจำลองของสิ่งที่มีจริง เช่น ในกรณีที่ผมกำหนดออปเจ็คเป็นแฟ้มข้อมูลเพื่อจะให้เป็นแบบจำลองของแฟ้มเก็บเอกสารใดก็ได้ สมมติว่าเป็นแฟ้มประวัติพนักงาน เมื่อเรามองในชีวิตจริงแฟ้มเอกสารจะมีชื่อแฟ้ม (แฟ้มประวัติพนักงาน) ภายในแฟ้มจะเก็บแบบฟอร์มที่เรากรอกประวัติพนักงานไว้ (คนละ 1 แบบฟอร์ม)

การทำงานจะได้แก่ นำแบบฟอร์มของพนักงานคนใหม่มาเก็บ ค้นหาแล้วนำแบบฟอร์มเก่าไปใช้งานหรือแก้แล้วนำมาเก็บ หรือถอนแบบฟอร์มออกจากแฟ้ม (เช่น ลาออก) ซึ่งอาจจะถอนทิ้งไปเลย หรือนำไปเก็บไว้ในแฟ้มประวัติศาสตร์

เมื่อเรานำมาทำเป็นแฟ้มข้อมูล ซึ่งจัดเป็นแบบจำลองของแฟ้มเก็บเอกสาร และเมื่อมองในฐานะออปเจ็คจะได้ว่ามีรายละเอียดประกอบด้วยชื่อแฟ้มและตัวแฟ้มซึ่งเป็นไฟล์ของเรคอร์ด เรคอร์ดหนึ่งคือแบบฟอร์มหนึ่ง สำหรับกิจกรรมคงเลียนแบบการทำงานดังกล่าวมา มีรายละเอียดดังจะกล่าวในสกุล DataFile

อนึ่ง ไฟล์ก็คือแฟ้มและแฟ้มก็คือไฟล์ แต่ในที่นี้คำว่าไฟล์ผมขอหมายถึงไฟล์ในดิสก์ ส่วนคำว่าแฟ้มขอหมายถึงแฟ้มเอกสารที่จับต้องได้ เว้นคำว่าแฟ้มข้อมูลซึ่งเป็นแบบจำลองของแฟ้มเอกสารจะเป็นไฟล์ในดิสก์

สกุล DataFile

จากการกำหนดออปเจ็คแฟ้มข้อมูล (data file) ได้นำมาสร้างเป็นสกุล DataFile ซึ่งจะเป็นการกำหนดตัวแปรไว้รองรับออปเจ็คและใช้ในโปรแกรมกับการกำหนดกิจกรรม กล่าวคือ

ตัวแปร

ตามรูปที่ 2 ได้แก่

– finame เก็บชื่อไฟล์ ซึ่งก็คือชื่อแฟ้มข้อมูลดังกล่าวมา ที่ให้เป็นสตริง 79 อักขระ เพื่อให้เก็บไดเรคทอรีย่อยด้วย

– dfi คือไฟล์ตามชื่อไฟล์

– fipstn เก็บตำแหน่งของไฟล์พอยน์เตอร์ที่ได้จากการค้นหา (search) เรคอร์ดที่ต้องการ

– recsi เก็บขนาดของเรคอร์ด

– onrec เป็นบัฟเฟอร์เพื่อเก็บข้อมูล 1 เรคอร์ด สำหรับการดำเนินการเป็นฟิลด์

– buff เป็นบัฟเฟอร์สำหรับเก็บข้อมูลทุกเรคอร์ด หรือเท่าที่จะมีหน่วยความจำเหลือใช้ในการค้นหาข้อมูล

– fldalt คือโครงสร้างของเรคอร์ด

– numrec เก็บจำนวนเรคอร์ดในไฟล์ขณะนั้น

– buffsi เก็บขนาดหน่วยความจำของ buff

– chcntr นับจำนวนอักขระจาก

```
UNIT  DATFI;

Interface

USES  crt,prime,rcrdconv;

CONST   MAXBUFF = 65535;

TYPE

    DataFile = OBJECT
      finme   : Str79;
      dfi     : File;
      fipstn  : LongInt;
      recsi   : Word;
      onerec,buff : PtrAryChr;
      fldatt : PtrAryRecFldAtt;
      numrec,buffsi : LongInt;
      chcntr,rpb,tme,offset,reclded : Word;
      frstsrch : Boolean;
```

รูปที่ 2

ต้น buff

– rpb ย่อมาจาก record per buff คือจำนวนเรคอร์ดใน 1 บัฟเฟอร์ buff

– recIded คือจำนวนเรคอร์ดที่โหลดได้จริงในบัฟเฟอร์ buff

– tme จำนวนครั้งที่จะอ่านข้อมูลจากไฟล์ลงบัฟเฟอร์ buff หมด

– offset คือออฟเซตของฟิลด์นับจากต้นเรคอร์ด

– frstsrch แสดงสถานะว่าเป็นการค้นหาครั้งแรกหรือไม่เพื่อการค้นหาต่อเนื่อง

กิจกรรมเกี่ยวกับไฟล์

ผมควรจะกล่าวถึงกิจกรรมหลักก่อน แต่เพื่อความต่อเนื่องจึงขอนำกิจกรรมเกี่ยวกับไฟล์และ Init มากล่าวถึงก่อน ตามรูปที่ 3 คือ

● **Init** เพื่อเตรียมค่าต่าง ๆ ได้แก่

- ให้ dfi แทนไฟล์ใน finame (ให้ค่าแก่ finame ด้วย) และเปิดใช้ไฟล์นี้ หากไม่มีไฟล์ในดิสก์จะสร้างเป็นไฟล์ว่างให้ด้วย Rewrite

- ให้ numrec เท่ากับจำนวนเรคอร์ดในไฟล์

- จองหน่วยความจำให้แก่ onerec มีขนาดเท่ากับ recsi ซึ่งได้ค่ามาจาก recsize สำหรับ buff ให้เป็น nil

- ให้ค่า fieldalt แก่ fldalt

- ให้ค่าเริ่มต้นแก่ตัวแปรต่าง ๆ ตามที่กล่าวมา

● **ResetDfi** ใช้แทนคำสั่ง Reset เพื่อช่วยให้ผู้เขียนโปรแกรม ไม่ต้องจำชื่อตัวแปรไฟล์ dfi และเขียนสื่อความได้สั้นขึ้นคือไม่ต้องเขียนเป็น

```
Reset (dfi, recsi)
```

● **CloseDfi** ใช้แทนคำสั่ง Close เพื่อช่วยผู้เขียนโปรแกรมคนต่อไปเช่นกัน

● **SeekDfi** เลื่อนไฟล์พอยน์เตอร์ของไฟล์และ fipstn ไปยังตำแหน่ง pstn

● **EofDfi** ใช้แทนฟังก์ชัน EOF (dfi)

การดำเนินการของกิจกรรมดังกล่าวมานี้ หากมีข้อผิดพลาดเกิดขึ้นจะให้ข้อผิดพลาดไว้ใน errnum (อยู่ในยูนิต prime) และมอบให้ผู้เขียนโปรแกรมคนต่อไปเป็นผู้พิจารณาว่าจะตรวจสอบหรือไม่

กิจกรรมหลัก

เพื่อเลียนแบบการทำงานด้วยมือดังกล่าวมาอยู่ในรูปที่ 4 ได้แก่

● **Search** ค้นหาเรคอร์ดที่ต้องการตามคีย์เวอร์กใน key ซึ่งอยู่ที่ฟิลด์ fld ได้ค่าใน onerec และให้ fipstn ชี้ ณ ตำแหน่งของเรคอร์ดนี้ เทียบได้กับที่ค้นหาแบบฟอร์มที่ต้องการโดยค้นที่ชื่อหรือหมายเลขโทรศัพท์ เป็นต้น

● **SrchCont** ในการค้นหาข้อมูลด้วย Search เช่น ใช้ชื่อเป็นคีย์เวิร์ด ไม่จำเป็นต้องใช้ชื่อและนามสกุลเต็มเป็นคีย์ อาจให้คีย์เป็นชื่อพยางค์แรกหรืออักษรตัวแรกได้ และถ้าไม่แน่ใจอักษรตัวใดอาจใช้เครื่องหมาย ? แทนได้ ซึ่ง Search จะให้ผลเป็นเรคอร์ดแรกที่เทียบตรงกันและยุติการค้นหา หากต้องการค้นต่อเนื่องจนกว่าจะพบบุคคลที่ต้องการจะต้องเปลี่ยนเป็นใช้ SrchCont (ตั้งแต่เริ่มแรก) และจะค้นพบหรือไม่ก็ตามจะต้องยุติการค้นหาด้วย CeaseSrch เสมอ

● **ReadRec** อ่านข้อมูล ณ ตำแหน่ง fipstn ให้แก่ rcrd เทียบได้กับที่หยิบแบบฟอร์มออกจากแฟ้มไปใช้งาน (อ่านหรือแก้)

● **WriteRec** เขียนข้อมูลตาม rcrd ลงในไฟล์ ณ ตำแหน่ง fipstn เทียบได้กับที่นำแบบฟอร์มมาเก็บในแฟ้มตาม

```
constructor  Init(finame : Str79;
                  fieldatt : Pointer; recsize : Word);
procedure    ResetDfi;
procedure    CloseDfi;
procedure    SeekDfi(pstn : LongInt);
function     EofDfi : Boolean;
```

รูปที่ 3

```
procedure    DelRec;
procedure    AddRec(rcrd : Pointer);
procedure    ReadRec(rcrd : Pointer);
procedure    WriteRec(rcrd : Pointer);
procedure    ReadCont(rcrd : Pointer);
procedure    SwapRcrd(frst,scnd : LongInt);

function     Search(var key; fld : Word) : Boolean;
function     SrchCont(var key; fld : Word) : Boolean;
procedure    GetFld(fld : Word; var dat);
procedure    StoreFld(fld : Word; var dat);
procedure    SaveOldRec;
procedure    CeaseSrch;
procedure    OneBuffSrch(var dat; tpe : Char;
                         var res : CompT); virtual;
```

รูปที่ 4

เดิม แต่ในกรณีของไฟล์ถ้าไม่แก้ไม่ต้องนำมาเก็บ

● **AddRec** เก็บข้อมูลตาม rcrd ต่อท้ายไฟล์โดยเลื่อน fipstn ไปท้ายไฟล์แล้วเรียกใช้ WriteRec

● **DelRec** ถอนข้อมูล ณ ตำแหน่งที่ fipstn ออกจากไฟล์ โดยนำข้อมูลที่อยู่ท้ายไฟล์มาแทนที่และลดค่า numrec ลงหนึ่ง

● **GetFld**

● **StoreFld**

สองกิจกรรมที่ดำเนินการกับฟิลด์ใน onerec เช่น หลังจากใช้คำสั่ง Search โดยที่ GetFld จะอ่านข้อมูล ณ ฟิลด์ที่ fld ให้แก่ dat ส่วน StoreFld จะนำข้อมูลตาม dat ไปเก็บ ณ ฟิลด์ที่ fld (ใน onerec ไม่ใช่ในไฟล์) ขอให้สังเกตการกำหนดตัวแปร dat ด้วยครับ ว่าไม่มีแบบข้อมูล เป็นหน้าที่ของผู้เขียนโปรแกรมคนต่อไปที่จะต้องผ่านค่าตัวแปรที่ตรงกับแบบข้อมูลของฟิลด์นั้นมาให้ เช่น เป็นสตริงหรือเลขจำนวนเต็ม

● **SaveOldRec** เก็บข้อมูลตาม onerec ลงไฟล์ ณ ตำแหน่ง fipstn จุดประสงค์คือเพื่อใช้กับ GetFld และ StoreFld คือเมื่อแก้ข้อมูลใน onerec ด้วย StoreFld แล้วจะต้องเก็บลงไฟล์ด้วย SaveOldRec ตัวอย่างเช่น

ในโครงงานห้องสมุดเมื่อมีผู้มายืมหนังสือ จะค้นหาหนังสือเล่มนั้นด้วยเลขเรียกหนังสือ (callnum เช่น ISBN...) เมื่อค้นพบจะได้รายละเอียดของหนังสือเล่มนั้นใน onerec อ่านฟิลด์ remain (หนังสือคงเหลือ) ว่ามีหนังสือให้ยืมหรือไม่ หากมีหนังสือให้ยืมก็จะเพิ่มหนึ่งค่าใน borrow (จำนวนหนังสือที่มีผู้ยืม) นำไปเก็บใน onerec และลดหนึ่งค่าใน remain นำไปเก็บใน onerec แล้วจึงเก็บลงไฟล์

● **Fin** เพื่อยกเลิกหน่วยความจำในฮีพที่จองไว้ และปิดไฟล์ที่เปิดค้างอยู่

ในการเขียนโปรแกรมตามแบบเดิม มีคำแนะนำว่าเมื่อเปิดไฟล์ใช้งานแล้วให้ปิดไฟล์ทุกครั้ง ทั้งนี้เพราะว่าถ้าเป็นการเปิดไฟล์เพื่อการอ่าน หากเปิดทิ้งไว้มาก ๆ อาจเกินจำนวนที่กำหนดไว้ แต่ถ้าเป็นการเปิดไฟล์เพื่อการเขียน นอกจากผลเสียดังกล่าว หากจบโปรแกรมโดยลืมปิดไฟล์อาจมีข้อมูลส่วนหนึ่งตกค้างอยู่ในหน่วยความจำไม่ได้เก็บลงดิสก์

สำหรับการเขียนโปรแกรมในแบบโอโอพี และจากโปรแกรมของกิจกรรม Fin ดังกล่าวมานี้ หากถือเป็นหลักปฏิบัติว่าจะเริ่มใช้อินสแตนซ์ของสกุล DataFile ด้วย Init และจบการใช้ด้วย Fin ก็ไม่จำเป็นต้องปิดไฟล์ทุกครั้งที่เขียนกับไฟล์ เว้นแต่เมื่อต้องการเปิดไฟล์เป็นจำนวนมากผลัดกันใช้งาน

กิจกรรมประกอบ

กิจกรรมที่จะกล่าวในหัวข้อนี้ หากคือหลักการเขียนโปรแกรมในแบบโอโอพีโดยเคร่งครัดแล้วจะต้องไม่กำหนดไว้เพราะยังไม่มีที่ใช้ แต่ผมเห็นว่าควรกำหนดไว้เพื่ออำนวยความสะดวกแก่ผู้เขียนโปรแกรมคนต่อไป ในรูปที่ 5 คือ

● **DelSrchRec**

● **WriteSrchRec**

● **ReadSrchRec**

จะค้นหาข้อมูลในไฟล์ด้วย Search ตามคีย์เวิร์ดใน key (ซึ่งจะต้องให้เป็นคีย์เต็ม เช่น ชื่อ นามสกุลเต็ม) ของฟิลด์ที่ fld เมื่อพบให้ถอนเขียนทับหรืออ่านด้วย DelRec, WriteRec และ ReadRec ตามลำดับ

● **ReadCont** อ่านข้อมูล ณ ตำแหน่งไฟล์พอยน์เตอร์ (ไม่ใช่ fipstn) ให้แก่ rcrd จึงทำให้สามารถอ่านต่อเนื่องไปจนสุดไฟล์ได้

โปรแกรมย่อย

เป็นกิจกรรมภายในของสกุลตามรูปที่ 4 ที่ไม่ต้องการให้ผู้เขียนโปรแกรมคนต่อไปนำไปใช้ ซึ่งปกติผมจะงดให้คำอธิบาย แต่สำหรับกรณีนี้เพื่อความเข้าใจในการอ่านโปรแกรมจึงขอนำมาอธิบาย คือ

● **PrepSrch** ก่อนการค้นหาไม่ว่าจะเป็น Search หรือ SrchCont จะเรียกใช้กิจกรรมนี้เพื่อ

- รีเซตไฟล์ ทั้งเพื่อเก็บข้อมูลที่ค้างอยู่ลงไฟล์ และเพื่อเลื่อนไฟล์พอยน์เตอร์และ fipstn ไปต้นไฟล์
- จองเนื้อที่ในฮีพให้แก่ buff เท่ากับขนาดของไฟล์เป็นไบต์ (จำนวนเรคอร์ด × ขนาดของเรคอร์ด) หรือ

```
procedure  DelSrchRec(var key; fld : Word;
                      var ok : Boolean);
procedure  WriteSrchRec(var key,rcrd; fld : Word);
function   ReadSrchRec(var key,rcrd; fld : Word) : Boolean;

destructor  Fin;
procedure  CalOffSet(var fld,ofset : Word);
procedure  PrepSrch;
procedure  DoSrch(var key; fld : Word; var res : CompT);

           END;
```

รูปที่ 5

เท่ากับขนาดของหน่วยความจำที่เหลือหรือเท่ากับ 64 กิโลไบต์ แล้วแต่จำนวนไหนจะน้อยกว่ากัน (การอ่านข้อมูลด้วย BlackRead อ่านได้ไม่เกินครั้งละ 64 กิโลไบต์) ซึ่งต่อไปจะขอเรียกขนาดหน่วยความจำนี้ว่า 1 บัฟฟ์

- คำนวณจำนวนเรคอร์ดต่อ 1 บัฟฟ์ให้แก่ rpb และจำนวนครั้งที่จะต้องโหลดไฟล์ลง buff ให้แก่ tme

● **DoSrch** ในการค้นหาข้อมูลทั้ง Search และ SrchCont จะเรียกใช้ DoSrch ดำเนินการให้ซึ่งจะ

- โหลดข้อมูลจากไฟล์ 1 บัฟฟ์ให้แก่ buff

- เรียกใช้ OneBuffSrch เพื่อค้นหาข้อมูลใน 1 บัฟฟ์

- หากพบจะยุติการค้นและกลับไปยังโปรแกรมที่เรียกมา (แต่คงค่าต่าง ๆ ไว้เพื่อการค้นต่อ) ได้ผลการค้นใน res เป็น EQ

- หากไม่พบจะโหลดข้อมูลให้แก่ buff อีก 1 บัฟฟ์และให้ OneBuffSrch ค้นต่ออีกจนกว่าจะโหลดข้อมูลจนหมด ได้ผลการค้นใน res เป็น GT หรือ LT ตามผลของการเทียบข้อมูลครั้งสุดท้าย

● **OneBuffSrch** ค้นหาข้อมูลใน 1 บัฟฟ์ โดยเรียกใช้ CompData (อยู่ในยูนิต prime) ให้เทียบข้อมูลของฟิลด์ในเรคอร์ดให้ได้ผลการเทียบใน res นำมาดำเนินการต่อคือ

- ถ้าพบคือ res มีค่าเป็น EQ จะลอกข้อมูลให้แก่ onerec และกลับไป ซึ่งค่าใน fipstn จะเป็นค่าตำแหน่งของเรคอร์ดนี้ในไฟล์ด้วย

- ถ้าไม่พบจะให้เทียบในเรคอร์ดต่อ ๆ ไปจนสุดบัฟเฟอร์

● **CalOffSet** คำนวณค่าออฟเซตของฟิลด์ที่ fld จากต้นเรคอร์ดให้แก่ ofset

```
CONST
  adrfld : Str19 = 's10s40s70s20s10s12';
```

รูปที่ 6 สตริงแบบข้อมูล

โปรแกรมทดสอบ

ผมได้เขียนโปรแกรมทดสอบการทำงานของสกุล DataFile ไว้ในไฟล์ TSTDATFI.PAS เพื่อแสดงการเขียนอ่านและแก้ข้อมูลในไฟล์ด้วยกิจกรรมตามที่กล่าวมา และเพื่อแสดงให้เห็นว่าเราสามารถดำเนินการกับข้อมูลเป็นเรคอร์ดได้โดยไม่ต้องกำหนดแบบข้อมูลเรคอร์ดและตัวแปรเรคอร์ด คือเพียงแค่กำหนดแบบข้อมูลเป็นสตริงแบบข้อมูลดังนำมาแสดงไว้ในรูปที่ 6

สำหรับการค้นหาข้อมูลตามที่กล่าวมานี้ เป็นการค้นในแบบ sequential ซึ่งจะใช้เวลาในการค้นหามาก หากต้องการให้ค้นได้เร็วต้องทำเป็นอินเด็กซ์ เช่น ทำเป็นแฟ้มดัชนีข้อมูล คือ ทำเป็นสกุล IdxDataFi เป็นผู้สืบสกุลของ DataFile อีกทอดหนึ่ง

```
NewOne = OBJECT
  perfrm  : SimForm;
  perfi   : DataFile;
  ..................
          END;

NewTwo = OBJECT(SimForm)
  perfi   : DataFile;
  ..................
          END;

NewThree = OBJECT(DataFile)
  perfrm  : SimForm;
  ..................
          END;
```

รูปที่ 7 แสดงการกำหนดสกุลใหม่ในการรวมสกุล

สกุลรวม

แฟ้มข้อมูลคือ สกุล DataFile เป็นแบบจำลองของแฟ้มเอกสารดังที่กล่าวมาคือ เราสามารถเปิดแฟ้ม เลือกเอกสารออกมาใช้งาน แก้ นำไปเก็บรวมทั้งสามารถถอดถอนเอกสารได้ด้วย ดังที่ยกการยืมหนังสือมาเป็นตัวอย่าง

แต่ยังไม่สมบูรณ์ที่เดียวนักคือ ยังขาดการติดต่อกับผู้ใช้ที่จะให้ผู้ใช้เลือกดำเนินการใด ๆ ได้ด้วยตนเอง ผมได้ออกแบบไว้แล้วว่าจะให้ใช้งานร่วมกับแบบกรอกข้อมูลคือสกุล SimForm การใช้งานร่วมนี้ทำได้หลายแบบ แบบหนึ่งคือไม่กำหนดสกุลใหม่เขียนเป็นโปรแกรมใช้งานเลย แต่มีงานเป็นจำนวนมากที่มีลักษณะเหมือนกันไม่ต้องการการดำเนินการใดเป็นพิเศษ เช่น พวกทะเบียนทั้งหลาย ดังเช่น ทะเบียนสมาชิก ทะเบียนหนังสือ ทะเบียนรถยนต์ เป็นต้น สมควรที่จะสร้างเป็นสกุลใหม่โดยรวมสกุลทั้งสองนี้เข้าด้วยกัน ซึ่งทำได้ 3 แบบดังแสดงในรูปที่ 7 คือ

- แบบแรก เป็นการกำหนดสกุล NewOne ขึ้นใหม่โดยมีอินสแตนซ์ของสกุลทั้งสองอยู่ในสกุลใหม่นี้ การกำหนดสกุลแบบนี้มีลักษณะเช่นเดียวกับการเขียนโปรแกรมเป็นยูนิต เพียงแต่เปิดโอกาสให้กำหนดอินสแตนซ์ของสกุลใหม่ได้เท่านั้น

- แบบที่สอง ให้ NewTwo เป็นผู้สืบสกุลของ SimForm โดยมีอิน-

สแตนซ์ของ DataFile อยู่ในสกุล

- แบบที่สาม ให้ NewThree เป็นผู้สืบสกุลของ DataFile โดยมีอินสแตนซ์ของ SimForm อยู่ในสกุล

สองแบบหลังนี้เป็นไปตามข้อกำหนดของเทอร์โบปาสคาลที่ให้สืบสกุลได้จากสกุลเดียวเท่านั้น (ในเทอร์โบ C++ ให้สืบจากหลายสกุลได้) จึงต้องเลือกระหว่างสองแบบหลังนี้โดยมีข้อพิจารณาว่า

- ให้สกุลใหม่เป็นผู้สืบสกุลของสกุลที่มีแนวโน้มว่า จะต้องมีการเปลี่ยนแปลงต่อเติมมากกว่า สำหรับกรณีนี้จะได้แก่สกุล DataFile

- จากการออกแบบหากมีลักษณะเป็นออปเจ็ค คือมีลักษณะเป็นการใช้งานตัวแปรคือเป็นอินสแตนซ์ของสกุลมากกว่าจะเป็นบรรพบุรุษ ให้กำเนิดเป็นอินสแตนซ์ของสกุล ในกรณีนี้บังเอิญได้แก่สกุล SimForm พอดี ซึ่งถ้าหากต้องการใช้อินสแตนซ์ของสกุล DataFile ด้วย ก็จะเป็นการรวมสกุลตามแบบแรก

ดังนั้นจะสร้างสกุลใหม่คือ FormFi1 ให้เป็นผู้สืบสกุลของสกุล DataFi1 ที่ให้ชื่อว่า FormFi1 คือมีเลข 1 ต่อท้าย เพราะผมยังเขียนโปรแกรม FormFi ตัวจริงไม่เสร็จ ยังขาดโปรแกรมสนับสนุนบางตัว

สกุล FormFi1

ดังที่กล่าวแล้วว่าสกุล DataFile สมบูรณ์ในตัว แต่ขาดแค่เพียงการติดต่อกับผู้ใช้ ดังนั้นจุดประสงค์หลักของการสร้างสกุลใหม่นี้คือเพื่อการติดต่อกับผู้ใช้ จึงต้องดูว่าผู้ใช้ต้องการดำเนินการอย่างไรบ้างกับแฟ้มข้อมูล จะได้ว่าผู้ใช้ต้องการ

- ป้อนข้อมูลใหม่
- ค้นหาข้อมูลในไฟล์ตามคีย์เวิร์ดที่ต้องการเพื่อ
  - อ่าน
  - แก้
  - ถอนข้อมูลจากไฟล์

ในการป้อนข้อมูลใหม่ต้องการแบบกรอกข้อมูลธรรมดา สำหรับการค้นหาข้อมูลผู้ใช้อาจไม่ให้คีย์เวิร์ดเต็ม เช่น ในการค้นหาข้อมูลโดยใช้ชื่อ นามสกุลเป็นคีย์เวิร์ด ผู้ใช้อาจให้แต่ชื่อซึ่งจะซ้ำกันหลายเรคอร์ด เราต้องแสดงชื่อนามสกุลที่ซ้ำกับคีย์ทั้งหมด (และรายละเอียดอื่นตามที่ผู้เขียนโปรแกรมคนต่อไปจะเห็นสมควร) ให้เลือกอีกครั้ง ซึ่งควรให้แสดงบนจอภาพเรคอร์ดละบรรทัด (อาจไม่ครบทุกฟิลด์) นั่นคือต้องการแบบกรอกข้อมูลแบบตารางเพื่อใช้ในการแสดงผล

แบบกรอกข้อมูลทั้งสองชนิดนี้ต่างก็เป็นกิจกรรมในสกุล SimForm ปัญหาของเรา (สมมติว่ายังไม่ตัดสินใจดังกล่าวมา) คือจะให้สกุลใหม่นี้เป็นผู้สืบสกุลของ SimForm หรือจะทำเป็นอินสแตนซ์ของสกุล SimForm ไว้ในสกุลใหม่ และถ้าเป็นอินสแตนซ์จะให้มีกี่ตัว

จากตัวอย่างที่ยกมาสมมติว่ามีชื่อซ้ำกัน 10 คน เราจะต้องแสดงในแบบตารางทั้ง 10 คนให้ผู้ใช้เลือก เมื่อผู้ใช้เลือกเรคอร์ดใดจะต้องแสดงข้อมูลตามแบบกรอกข้อมูลธรรมดา เพื่อให้เป็นข้อมูลทุกฟิลด์แล้วให้ผู้ใช้เลือกดำเนินการ (แก้หรือถอนข้อมูลออกจากไฟล์) หลังจากนี้ผู้ใช้อาจต้องการไปตรวจรายชื่อในตารางเพื่อดำเนินการอย่างหนึ่งอย่างใดซ้ำเดิมอีก นั่นคือเราต้องการใช้สกุล SimForm ในฐานะออปเจ็คในสกุลใหม่ โดยกำหนดเป็นอินสแตนซ์ 2 ตัว

นอกจากนี้ในการออกแบบโปรแกรมที่ผมตัดตอนมาตั้งใจไว้ว่า ไม่ว่าผู้ใช้จะทำงานอะไรอยู่ให้สามารถผละจากงานนั้นชั่วคราวเพื่อไปบันทึกข้อความอะไรก็ได้ในหน้าต่าง NOTE PAD ขอให้ดูในรูปที่ 8 ครับ ซึ่งโปรแกรมหลักจะเป็นผู้เตรียมหน้าต่างนี้ (เป็นหน้าต่างในแบบ pop up นั่นคือ จะเก็บภาพพื้นหลังไว้ด้วย ทำให้สามารถแสดงทับไปบนหน้าต่างใดก็ได้ เมื่อให้เก็บหน้าต่าง

```
ADDRESS ENTER FORM

                    ADDRESS_ID

NAME

HOUSE NUMBER LANE AND
STREET

CITY                          ZIP                PHONE

F1=Note   F2=Save   F3=Backward   F4=Forward   F5=Exit

NOTE PAD
Delta One Charlie One Foxtrot One   Oscar Two   Kilo Two Foxtrot Two
Alpha One    Echo One    Kilo One   Alpha Two   Bravo One   Delta Two
Bravo Two Alpha three    Test One    Test Two
```

รูปที่ 8 แสดงภาพบนจอภาพ ขณะใช้งาน FomFi1

ก็จะได้ภาพพื้นเดิมคืนมา สกุลของหน้าต่างแบบนี้คือ PopWnd ซึ่งผมสร้างเพิ่มไว้ในยูนิต SMPLWND แต่ที่นำมาใช้นี้ไม่ใช่เพื่อการแสดงทับ หาก (เพื่อให้สามารถเก็บภาพในหน้าต่างของตน เมื่อมีการสับเปลี่ยนหน้าต่างอื่น) จึงต้องเตรียมการทำงานสัมพันธ์กับหน้าต่างนี้

จากที่กล่าวมานี้จะนำไปสร้างสกุล FormFil คือกำหนดตัวแปรกิจกรรมและเขียนโปรแกรมของกิจกรรมได้ตามรูปที่ 9 คือ

ตัวแปร

– fixfrm และ tabfrm เป็นอินสแตนซ์ของสกุล SimForm เพื่อเป็นการกรอกข้อมูลธรรมดาและตารางตามลำดับ

– notewnd เป็นพอยน์เตอร์ไปยังอินสแตนซ์ของสกุล SimWnd ซึ่งหมายถึงของสกุลลูกหลานต่อไปด้วย สำหรับกรณีนี้จะเป็นอินสแตนซ์ของสกุล PopWnd

– curfrm เป็นพอยน์เตอร์ไปยัง fixfrm หรือ tabfrm สุดแต่ว่าขณะนั้นใช้ฟอร์มใดอยู่ใช้งานคู่กับ notewnd

– userwant เป็นเซตของอักขระที่ผู้เขียนโปรแกรมคนต่อไปต้องการว่าหากผู้ใช้กดคีย์เหล่านี้คีย์ใดคีย์หนึ่งให้ยุติการทำงานในสกุลนี้ชั่วคราว

– frmkey คือ userwant ที่เพิ่มค่าคีย์หน้าที่พิเศษ F1 ถึง F4 เพื่อให้ออกจาก SimForm ชั่วคราว เพื่อจะให้ดำเนินการดังที่จะกล่าวต่อไป

กิจกรรมหลัก

สกุลนี้มีกิจกรรมหลักเพียง 2 กิจกรรมคือ EnterData กับ ScanData และได้รวมกิจกรรมที่ใช้ในการเตรียมการและเลิกการใช้งานสกุลนี้มาไว้ในหัวข้อนี้ด้วยคือ

● Init

เพื่อการเตรียมค่าต่าง ๆ และเตรียม DataFile กับ fixfrm ขอให้สังเกตว่าได้ให้ผ่านค่าหน้าต่าง NOTE PAD มาให้แก่ notepad ซึ่งจะให้ค่าแก่ notewnd อีกทอดหนึ่ง

● InitScan

เนื่องจากพารามิเตอร์ที่ผ่านมาให้แก่ Init มีมาก ผมจึงแยกส่วนที่จะเตรียมค่าให้แก่ tabfrm ซึ่งใช้ในการแสดงข้อมูลจากการค้นหามาให้แก่ InitScan

● EnterData

เพื่อการป้อนข้อมูลตามแบบกรอกข้อมูลธรรมดาดังแสดงในรูปที่ 7 ซึ่งจะเรียกใช้ EnterFixData ในสกุล SimForm อีกทอดหนึ่ง แต่ในระหว่างที่ผู้ใช้ป้อนข้อมูลอยู่นี้ หากกดคีย์

– F1 จะไปทำงานในหน้าต่าง NOTE PAD เพื่อให้ผู้ใช้บันทึกข้อความใดก็ได้

– F2 เก็บข้อมูลที่ป้อนลงไฟล์ด้วย AddRec

– F3 จะแสดงข้อมูลของเรคอร์ดก่อนหน้านี้ (ในไฟล์) ตามแบบฟอร์มบนจอภาพ ทั้งนี้เพื่อผู้ใช้สงสัยว่าข้อมูลที่เพิ่งป้อนไปไม่ถูกก็ให้สามารถแก้ได้และต้องเก็บลงไฟล์ด้วย F2 จะกดคีย์ F3 ย้อนกลับไปกี่เรคอร์ดก็ได้ และให้กดคีย์ F4 ย้อนกลับมาทีละเรคอร์ด หรือกดคีย์ F5 เพื่อกลับมายังเรคอร์ดสุดท้ายโดยตรง

```
UNIT  FRMFIONE;

Interface

USES  crt,prime,smplwnd,rcrdconv,smplform,datfi;

TYPE

  FormFil = OBJECT(DataFile)
    fixfrm,tabfrm    : SimForm;
    notewnd,curfrm   : PtrSimWnd;

    userwant,frmkey : ChrSet;

  constructor  Init(finame : Str79; wndform : WndParmT;
                    formout,formin,rectype : Pointer;
                    keywant : ChrSet; notepad : PtrSimWnd);
  procedure  InitScan(wndfrm : WndParmT;
                      formout,formin,rectype : Pointer);
  procedure  OneBuffSrch(var dat; tpe : Char;
                         var res : CompT); virtual;

  procedure  EnterData;
  procedure  ScanData(var key; fld : Word; var chf : Char);

  destructor  Fin;
  procedure  ShowCurForm;
  procedure  HideCurForm;
  procedure  UsrNotePad(var chf : Char; kw : ChrSet);
  procedure  DoDelCrctRcrd(var chf : Char);
  procedure  DoBackWard(var chf : Char);
  procedure  DoFBwrd(disp : Boolean; var chf : Char);
  procedure  DispScanRcrd(var chf : Char);
             END;
```

รูปที่ 9 แสดงการกำหนดสกุล FormFil

– F5 หรือ Esc ยุติการป้อนข้อมูล

● **ScanData**

ค้นหาข้อมูลที่ผมเรียกเป็น scan เพราะเป็นการค้นต่อเนื่องจนกว่าจะได้ข้อมูลที่ตรงกับคีย์เวิร์ดทั้งหมด และกิจกรรมนี้ครับที่ผมต้องใช้ชื่อ FromFil แทนที่จะเป็น FormFi เพราะเมื่อค้นพบเวิร์ดหนึ่ง ๆ แล้วควรจะเก็บข้อมูลไว้ในอะเรย์ของเรคอร์ดก่อน ซึ่งควรเป็นอะเรย์ที่ไม่จำกัดจำนวนช่อง ดังที่ผมเคยทำเป็นสกุล Collection ไว้แต่ผมต้องการปรับปรุงสกุลนี้ก่อน

การทำงานของ ScanData จึงมีลักษณะเป็นเมื่อค้นพบเรคอร์ดหนึ่งแล้วให้นำออกแสดงด้วยแบบฟอร์มตาราง หากผู้ใช้ต้องการดูรายละเอียดให้กดคีย์ F2 จะแสดงด้วยแบบฟอร์มธรรมดาและให้ผู้ใช้เลือกอีกว่า ถ้ากดคีย์ F2 ให้ถอนเรคอร์ดนี้จากไฟล์ หากต้องการแก้ไขให้กดคีย์ F3 หากกดคีย์ F4 จะค้นหาเรคอร์ดต่อไปมาแสดงในแบบตารางอีกจนกว่าจะสุดไฟล์หรือยุติการทำงานด้วยคีย์ F5 หรือเมื่อผู้ใช้เลือกถอนหรือแก้ข้อมูลก็จะยุติการทำงานเช่นกัน

● **Fin**

เพื่อยุติการทำงาน

โปรแกรมย่อย

โปรแกรมย่อยที่สนับสนุนการทำงานของกิจกรรมหลักได้แก่

● **ShowCurForm**

● **HideCurForm**

เพื่อให้แสดงและเก็บฟอร์ม (ตามลำดับ) เป็นฟอร์มที่ให้ค่าไว้แก่ curfrm ทั้งนี้เพราะเป็นส่วนของโปรแกรมที่ค่อนข้างยาวดังรูปที่ 10 จึงทำเป็นโปรแกรมย่อย

● **UsrNotePad**

เปิดใช้หน้าต่าง NOTE PAD และกลับมายังฟอร์มที่ดำเนินการค้างอยู่

● **DoBackWard**

ในขณะที่ผู้ใช้ป้อนข้อมูลหากกดคีย์ F3 จะมาทำงานที่โปรแกรมย่อยนี้เพื่อเตรียมค่าต่าง ๆ แล้วเรียกใช้ DoFBwrd

● **DoFBwrd**

จะลดหนึ่งหรือเพิ่มหนึ่งค่าใน fipstn ตามแต่ว่าค่าที่ส่งมาให้แก่ chf และที่ผู้ใช้จะกดต่อไปเป็นคีย์ F3 หรือ F4 ตามลำดับ อ่านเรคอร์ด ณ ตำแหน่ง fipstn ออกแสดงและให้ผู้ใช้แก้ได้ด้วย Enter FixData ดังนั้นในเที่ยวแรก ค่า chf จะเป็น F3 เป็นการถอยหลัง หากผู้ใช้กดคีย์ F3 ก็จะถอยหลังอีก หากกดคีย์ F4 จะเป็นการเดินหน้า หากผู้ใช้กดคีย์ F5 จะออกจากโปรแกรมนี้

● **DispScanRcrd**

ในการค้นหาข้อมูลด้วย ScanData เมื่อพบเรคอร์ดที่ต้องการ จะมาทำงานที่โปรแกรมนี้นำข้อมูลออกแสดงในแบบตาราง และรอจนกว่าผู้ใช้จะกดคีย์ใน userwant หรือ F1, F2, F4 หรือ Enter ถ้าเป็น Enter จะให้ chf มีค่าเป็น F2

● **DoDelCrctRcrd**

ในการค้นหาข้อมูลหากผู้ใช้กดคีย์ F2 จะมาทำงานในโปรแกรมนี้เพื่อให้ผู้ใช้เลือกว่าจะแก้หรือถอนเรคอร์ด แล้วดำเนินการตามที่ผู้ใช้ต้องการ

โปรแกรมทดสอบ

ได้เขียนโปรแกรมสำหรับทดสอบการทำงานของโปรแกรมนี้ไว้ในรูปที่ 11 และ 12 โดยที่ในรูปที่ 11 มีการดำเนินการดังนี้

● ได้ include ไฟล์ 2 ไฟล์คือ

– addrfrm1.icl เก็บสตริงแบบข้อมูล อินพุตฟอร์มกับเอาต์พุตฟอร์ม และพารามิเตอร์ของหน้าต่าง ตามที่กล่าวในฉบับที่แล้ว

– notefrm.icl เก็บพารามิเตอร์ของหน้าต่าง

● ในการกำหนดตัวแปรได้กำหนดให้ perfile (หมายถึงแฟ้มบุคคล) ให้เป็นอินสแตนซ์ของสกุล FormFi และให้ note (คือ NOTE PAD) เป็นอินสแตนซ์ของสกุล PopWnd สำหรับ sltmem จะให้เก็บขนาดของหน่วยความจำที่ว่างในฮีพเพื่อเทียบในตอนท้ายว่าได้ยกเลิกหน่วยความจำที่จองไว้หมดจดหรือไม่

● โปรแกรมย่อย DispName จะอ่านข้อมูลที่มีอยู่ในไฟล์และนำเฉพาะฟิลด์ชื่อออกแสดง ทั้งเพื่อทดสอบการเขียนกับไฟล์และเพื่อแสดงว่าเราจะอ่านฟิลด์ของเรคอร์ดในไฟล์ได้อย่างไร

โปรแกรมในรูปที่ 12 เป็นโปรแกรมหลัก เริ่มด้วยการให้ค่าเริ่มต้นแก่อิน-

```
PROCEDURE  FormFil.ShowCurForm;
BEGIN
  if  notewnd = nil  then  begin  curfrm^.Show;  Exit  end;
  notewnd^.Hide;
  curfrm^.Show;  curfrm^.WndOff;
  notewnd^.Show;  notewnd^.WndOff;
  curfrm^.WndOn;
END;
```

รูปที่ 10 โปรแกรมย่อย ShowCurForm

สแตนซ์ทั้งสอง แล้วทดสอบกิจกรรม EnterData กับ ScanData ตอนท้ายเป็นการตรวจสอบหน่วยความจำว่าถูกต้องหรือไม่ หากต้องการทดสอบกิจกรรมใดเพิ่มขอให้ป้อนในช่วงที่ได้หมายเหตุไว้ว่า Put test program here

```
PROGRAM  TestFormFi;

USES  crt,prime,smplwnd,rcrdconv,smplform,datfi,frmfione;

{$I addrfrm1.icl}
{$I notefrm.icl}


VAR   perfile : FormFi1;  note : PopWnd;  sttmem : LongInt;
      name : Str39;  ok : Boolean;  ch : Char;


PROCEDURE  DispName;
VAR   cntr : Word;  nme : Str79;
BEGIN
  perfile.ResetDfi;  cntr := 0;  note.WndOn;
  while  not  perfile.EofDfi  do
    with  perfile  do
    begin
    ReadCont(onerec);  GetFld(2,nme);
    Write(nme:12);  inc(cntr);
    if  cntr mod 6 = 0  then  WriteLn;
    end;
  WriteLn;  note.WndOff;
END;
```

รูปที่ 11

```
BEGIN   { MAIN }
  sttmem := MemAvail;  ClrScr;
  note.Init(ntewndfrm);
  note.Show;

  with  perfile  do
    begin
    Init('PERSON.DTA',adrwndfrm,
         @adrOform,@adrIform,@adrfld,[],@note);
    InitScan(adrWndTab,@adrOtab,@adrItab,@adrTfld);
    DispName;

(* Put test program here *)

    EnterData;
    name := 'Alpha';  ScanData(name,2,ch);
    Write(' press Enter ');    ReadLn;

(* ..........  *)
    end;

perfile.Fin;
note.Hide;  note.Fin;
Window(1,1,80,25);  ClrScr;
WriteLn('At start  mem = ',sttmem);
WriteLn('At finish mem = ',MemAvail);
END.
```

รูปที่ 12

กับโอโอพี

หวังว่าคุณคงได้สัมผัสกับโอโอพีพอสมควร ขอให้ลองกำหนดเรคอร์ดด้วยตนเอง โดยกำหนดเป็นสตริงแบบข้อมูล และกำหนดเอาต์พุตฟอร์มกับอินพุต โดยกำหนดเป็นสตริงแบบข้อมูลและกำหนดเอาต์พุตฟอร์มกับอินพุตฟอร์ม ทดลองใช้งานสกุล FormFi1 ดูครับจะเห็นว่าใช้งานได้กว้างขวางทีเดียว แทบกล่าวได้ว่าไม่ต้องเขียนโปรแกรมสำหรับป้อน แก้ไข และค้นหาข้อมูลอีกเลย และถ้ามีผู้สร้างนามสกุลไว้สำหรับงานต่าง ๆ ต่อไปคงไม่ต้องเขียนโปรแกรมกัน (เขียนแต่โปรแกรมใช้งานสั้น ๆ)

และถ้าเราหวังพึ่งซอฟต์แวร์จากต่างประเทศ ปัญหาที่จะเกิดขึ้นก็คืออาจปรับปรุงใช้กับภาษาไทยไม่ได้ ดังโปรแกรมที่ผมนำมาเสนอนี้ (หากคุณมีแค่ไฟล์ .TPU) หรือในทางตรงข้ามอาจเลิกเขียนโปรแกรมกันไปเลย ในทัศนะของผมโอโอพีจึงมีคุณอนันต์โทษมหันต์